An introduction to searching in oXygen using XPath

James Cummings
@jamescummings

14 September 2014
XPath

XPath is the basis of most other XML querying and transformation languages (such as XSLT and XQuery). It is used in many other contexts (Editors, XML Databases, Content Management Systems, Configuration Files) as a way to say what bit(s) of an XML document you are using.

It is just a way of locating nodes in an XML document.
What does XPath do?

- XPath does nothing! It merely documents a set of nodes. An XPath processor does something with this.
- An XPath expression matches a set of nodes
- It encodes an 'address' for the selected nodes, sort of like a URL does or a file path
- This address can be given in a variety of ways including absolute or relative methods
<body n="anthology">
  <div type="poem">
    <head>The SICK ROSE</head>
    <lg type="stanza">
      <l n="1">O Rose thou art sick.</l>
      <l n="2">The invisible worm,</l>
      <l n="3">That flies in the night</l>
      <l n="4">In the howling storm:</l>
    </lg>
    <lg type="stanza">
      <l n="5">Has found out thy bed</l>
      <l n="6">Of crimson joy:</l>
      <l n="7">And his dark secret love</l>
      <l n="8">Does thy life destroy.</l>
    </lg>
  </div>
  <div type="shortpoem">
    <head>Amorous Rhyme</head>
    <lg type="couplet">
      <l>What inspired this amorous rhyme?</l>
      <l>Two parts vodka, one part lime</l>
    </lg>
  </div>
</body>
XML Structure

Really attributes (and
text) are separate nodes!
Basic XPath abbreviated syntax

/   Level selector
// Multi-level selector
../ one level up (relative)
*   Wildcard
@  Attribute prefix
@* Attribute wildcard
[...] Filter / condition

function(item, 'parameter') function with parameters

Every expression 'step' creates a new context, narrowing down the choice (usually) as you go further down the path.
XPath locates any matching nodes
/body/div/@type ?

@ = attributes

body type="anthology"

div type="poem"

div type="shortpoem"

head

lg type="stanza"

ln="1"

ln="2"

ln="3"

ln="4"

ln="5"

ln="6"

ln="7"

ln="8"

lg type="couplet"

ln="1"

ln="2"
Slide 14/47

Square Brackets Filter
Selection

/body/div/lg/l[@n="2"] ?

```plaintext
body type="anthology"
div type="poem"
div type="shortpoem"
head
lg type="stanza"
n="1"
n="2"
n="3"
n="4"

lg type="stanza"
n="5"
n="6"
n="7"
n="8"

lg type="couplet"
n="1"
n="2"
```
/body/div[@type="poem"]/head

body type="anthology"

div type="poem"

body type="anthology"

div type="shortpoem"

head

lg type="stanza"

ln="1"

ln="2"

ln="3"

ln="4"

ln="5"

ln="6"

ln="7"

ln="8"

lg type="couplet"

ln="1"

ln="2"
//lg[@type=“stanza”]  ?

// = any descendant

body type=“anthology”
  
  div type=“poem”
    
    div type=“shortpoem”
      
      lg type=“stanza”
        
        lg type=“stanza”
          
          head

lg type=“stanza”
  
  head

lg type=“couplet”
  
  l n=“1”
  l n=“2”

l n=“3”
  
  l n=“4”

l n=“5”
  
  l n=“6”

l n=“7”
  
  l n=“8”

l n=“1”
  
  l n=“2”
//lg/../@type ?

Paths are relative:
.. = parent
Numerical operations can be useful.
Notice the deleted <head>!
//div[head]/lg/l[@n="2"]
ancestor:: is an unabbreviated axis name
XPath: More About Paths

- A location path results in a node-set
- Paths can be absolute (/div/lg[1]/l)
- Paths can be relative (l/../../head)
- Formal Syntax: (axisname::nodetest[predicate])
- child::div[contains(lower-case(head), 'ROSE')]

Details:
- Location path results in a node-set.
- Paths can be absolute or relative.
- Formal syntax includes axisname, nodetest, and predicate conditions.
XPath: Axes

ancestor::: Contains all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self::: Contains the current node plus all its ancestors (parent, grandparent, etc.)
attribute::: Contains all attributes of the current node
child::: Contains all children of the current node
descendant::: Contains all descendants (children, grandchildren, etc.) of the current node
descendant-or-self::: Contains the current node plus all its descendants (children, grandchildren, etc.)
XPath: Axes (2)

- **following::** Contains everything in the document after the closing tag of the current node
- **following-sibling::** Contains all siblings after the current node
- **parent::** Contains the parent of the current node
- **preceding::** Contains everything in the document that is before the starting tag of the current node
- **preceding-sibling::** Contains all siblings before the current node
- **self::** Contains the current node
Axis examples

- ancestor::lg = all `<lg>` ancestors
- ancestor-or-self::div = all `<div>` ancestors or current
- attribute::n = n attribute of current node
- child::l = `<l>` elements directly under current node
- descendant::l = `<l>` elements anywhere under current node
- descendant-or-self::div = all `<div>` children or current
- following-sibling::l[1] = next `<l>` element at this level
- preceding-sibling::l[1] = previous `<l>` element at this level
- self::head = current `<head>` element
XPath: Predicates

- `child::lg[number(translate(@type, 'stanza', 'a') = 'a')]`
- `child::l[number(@n = '4')]`
- `child::div[position() = 3]`
- `child::div[4]`
- `child::l[last()]`
- `child::lg[last() - 1]`
XPath: Abbreviated Syntax

- nothing is the same as `child::`, so `lg` is short for `child::lg`
- `@` is the same as `attribute::`, so `@type` is short for `attribute::type`
- `.` is the same as `self::`, so `.//head` is short for `self::node()/child::head`
- `../` is the same as `parent::`, so `../lg` is short for `parent::node()/child::lg`
- `//` is the same as `descendant-or-self::`, so `div//l` is short for `child::div/descendant-or-self::node()/child::l`
XPath: Functions

XPath contains numerous functions: you've seen 'last()' already. There are also many other functions, including:

**Conversion:** boolean, string, number

**Contexts:** count, deep-equals, last, position

**DateTimes:** current-dateTime, day-from-dateTime, months-from-duration, timezone-from-dateTime

**Math:** avg, ceiling, count, floor, max, min, round, sum

**Logic:** true, false, not

**Nodes:** lang, local-name, name, namespace-uri, text

**Sequences:** distinct-values, empty, index-of, subsequence

**Strings:** concat, contains, lower-case, matches, normalize-space, replace, starts-with, string-length, substring, substring-after, substring-before, tokenize, translate, upper-case

and many more!
<TEI xmlns:teiext="http://www.tei-c.org/ns/1.0" xml:id="S2300" xmlns:teiheaders="">
  <fileDesc>
    <titleStmt>
      <titleSent>Protestant Cemetery, Rome: Stone S2300</titleSent>
      <publicationStmt>
        <p>Unpublished</p>
      </publicationStmt>
    </titleStmt>
    <sourceDesc>
      <msDesc facs="#lot_S2300">
        <msIdentifiers>
          <idno>S2300</idno>
        </msIdentifiers>
      </msDesc>
    </sourceDesc>
    <physDesc>
      <objectDesc form="HEAD">
        <supportDesc>
          <support>
            <materials>
              <material>stonework</material>
            </supports>
            <extent>
              <num-l=1/num-l>
                <dimensions unit="cm">
                  <height>62</height>
                  <depth>79</depth>
                  <width>21</width>
                </dimensions>
              </extent>
            </condition>good</condition>
          </supportDesc>
        </supportDesc>
      </objectDesc>
    </physDesc>
  </fileDesc>
</TEI>
Next we're going to have a bit of a demo of using XPath for searching in oXygen and do an exercise.

- XPath Tutorial: http://www.w3schools.com/xpath/
- XPath Functions: http://www.w3schools.com/xpath/xpath_functions.asp